Estructuras de Datos y Algoritmos

Segundo curso de los Grados en Ingeniería Informática

Examen Final, Septiembre de 2012.

1. **(3,5 puntos)** Especificar y derivar formalmente, o diseñar y verificar, un algoritmo iterativo de coste lineal, que reciba un vector no-vacío V de enteros y un entero N con la longitud de V, y calcule en una sola pasada el número de veces que aparece repetido su elemento menor.

2. **(3 puntos)** Suponemos declarado un tipo `personaje` con las operaciones:

```
esDragon   : personaje -> bool
esPrincesa : personaje -> bool
```

   Tenemos una variable del TAD `Arbin<personaje>`, cada uno de cuyos nodos puede ser un dragón, una princesa, o ninguno de los dos. Un nodo se dice *accesible*, si en el camino que lo une con la raíz no hay ningún nodo-dragón. Se pide:

   1. **(1 punto)** Implementar, como **usuaria del TAD**, una operación que devuelva el número de nodos-princesa accesibles
   2. **(2 puntos)** Implementar, como **usuaria del TAD**, una operación que devuelva la princesa accesible más cercana a la raíz. Se puede suponer que existe al menos una princesa accesible. El algoritmo **no debe inspeccionar** nodos que se encuentren a mayor profundidad que la princesa más cercana.

3. **(3,5 puntos)** Los ferrys son barcos que sirven para trasladar coches de una orilla a otra de un río. Los coches esperan en fila al ferry y cuando éste llega un operario les va dejando entrar.

   En nuestro caso el ferry tiene espacio para dos filas de coches (la fila de babor y la fila de estribor) cada una de N metros. El operario conoce de antemano la longitud de cada uno de los coches que están esperando a entrar en él y debe, según van llegando, mandarles a la fila de babor o a la fila de estribor, de forma que el ferry quede lo más cargado posible. Ten en cuenta que los coches deben entrar en el ferry **en el orden en que aparecen** en la fila de espera, por lo que en el momento en que un coche ya no entra en el ferry porque no hay hueco suficiente para él, no puede entrar ningún otro coche que esté detrás aunque sea más pequeño y sí hubiera hueco para él.

   Implementa una función que reciba la capacidad del ferry en metros y la colección con las longitudes de los coches que están esperando (puedes utilizar el TAD que mejor te venga) y devuelva la colocación óptima de los coches, entendiendo ésta como la secuencia de filas en las que se van metiendo los coches. Supón la existencia de los símbolos `BABOR` y `ESTRIBOR`.

   Ejemplo: Si el ferry tiene `5` metros de longitud y en la fila tenemos coches con longitudes (del primero al último) `2.5`, `3`, `1`, `1`, `1.5`, `0.7` y `0.8`, una solución óptima es `[BABOR, ESTRIBOR, ESTRIBOR, ESTRIBOR, BABOR, BABOR]`.